ELSEVIER

Computational Geometry 10 (1998) 203–220

Computational Geometry
Theory and Applications

Rectangular grid drawings of plane graphs

Md. Saidur Rahman [1], Shin-ichi Nakano *, Takao Nishizeki [2]

Graduate School of Information Sciences, Tohoku University, Sendai 980-77, Japan

Communicated by R. Tamassia; submitted 31 March 1997; accepted 7 January 1998

Abstract

The rectangular grid drawing of a plane graph G is a drawing of G such that each vertex is located on a grid point, each edge is drawn as a horizontal or vertical line segment, and the contour of each face is drawn as a rectangle. In this paper we give a simple linear-time algorithm to find a rectangular grid drawing of G if it exists. We also give an upper bound $W + H \leqslant n/2$ on the sum of required width W and height H and a bound $WH \leqslant n^2/16$ on the area of a rectangular grid drawing of G, where n is the number of vertices in G. These bounds are best possible, and hold for any compact rectangular grid drawing. © 1998 Elsevier Science B.V.

Keywords: Algorithm; Rectangular drawing; Grid drawing; Grid area; Floorplanning

1. Introduction

Recently automatic drawing of graphs has created intense interest due to its broad applications to represent various concepts and objects in software, computer architecture, networks, VLSI circuits, etc. [6]. One important criteria of a graph drawing is that it should be easily understandable and good looking [4,14,19]. In this paper we thus consider *the rectangular drawing* of a plane graph, where each edge of the graph is drawn as a horizontal or vertical line segment with no bend and the contour of each face is drawn as a rectangle as illustrated in Fig. 1(a). Our input graph G has four vertices of degree 2 on its outer face and all other vertices have degree 3. It is not always possible to draw such a graph in this way. For example, the graph in Fig. 2 has no rectangular drawing. Thomassen obtained a necessary and sufficient condition for such a plane graph to have a rectangular drawing [22]. His proof does not look to lead to an efficient algorithm for rectangular drawing; a straightforward implementation of his method requires at least $O(n^3)$ time, where n is the number of vertices in G.

On the other hand, Kozminski and Kinnen [12] established a necessary and sufficient condition for the existence of a "rectangular dual" of a plane graph, that is a rectangular drawing of the dual

* Corresponding author. E-mail: nakano@ecei.tohoku.ac.jp.

[1] E-mail: saidur@nishizeki.ecei.tohoku.ac.jp.

[2] E-mail: nishi@ecei.tohoku.ac.jp.

PII S0925-7721(98)00003-0

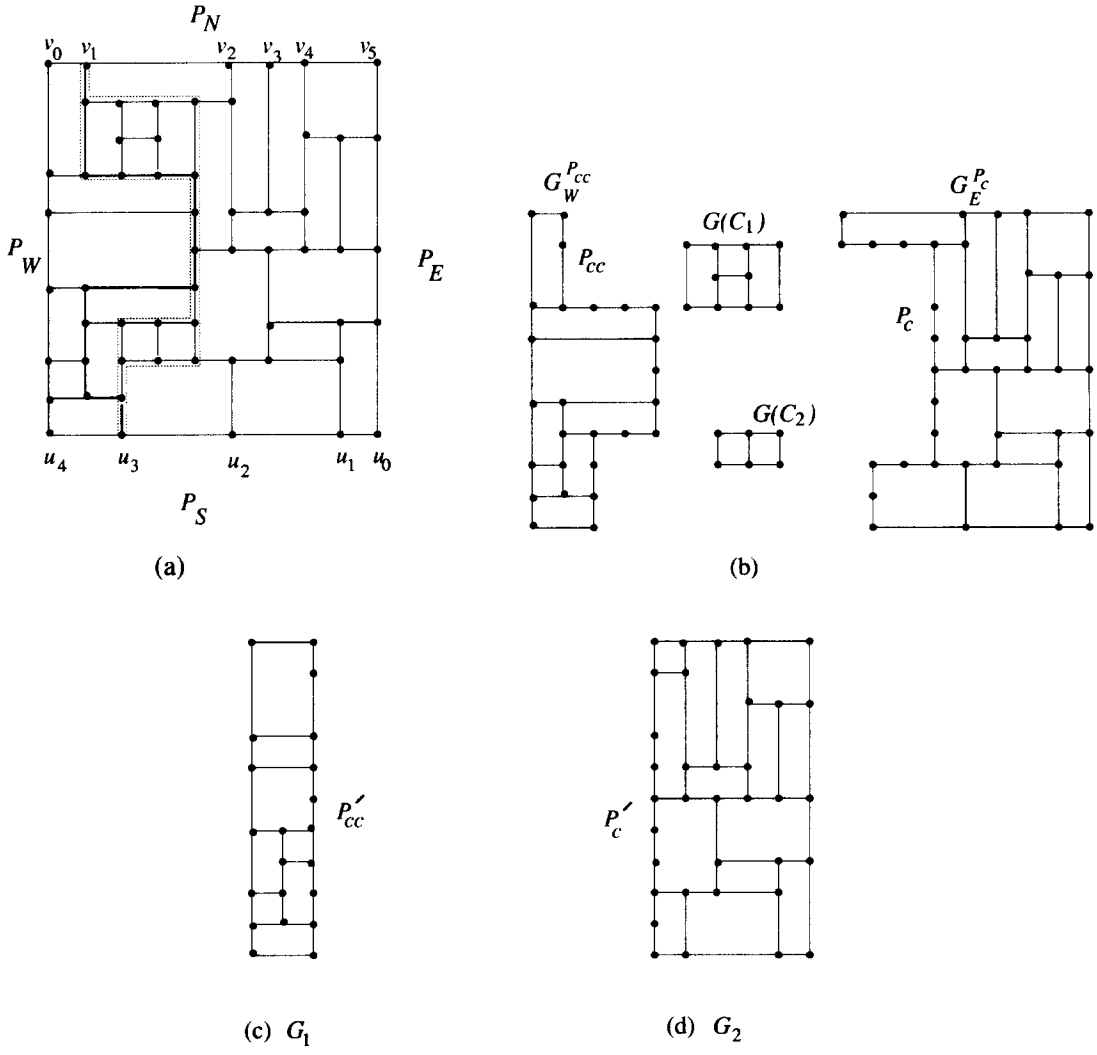

Fig. 1. A rectangular grid drawing of a plane graph and patching.

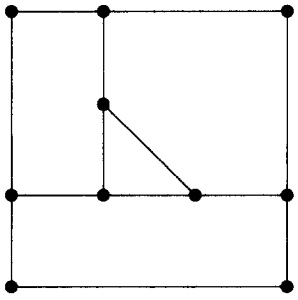

Fig. 2. A graph with no rectangular drawing.

graph of a given plane graph, and gave an $O(n^2)$ algorithm to obtain it. Since a rectangular dual has practical applications on VLSI floorplan [13,21], much attention has been paid to [12]. Based on the characterization of [12], Bhasker and Sahni [1] and He [8], respectively, developed linear-time algorithms to find a rectangular dual of a plane graph. Recently Kant and He [11] presented two more linear-time algorithms. The algorithm in [1] is fairly complicated and consists of two steps:
(1) finding a "regular edge labeling" of G, and
(2) constructing the rectangular dual using that labeling.
He [8] simplified step (2), but used the complicated algorithm in [1] for step (1). Two methods for simplifying step (1) are given in [11]. The first one finds a regular edge labeling using "edge contraction and edge expansion techniques" and is indeed not a simple method. The second one finds a "canonical ordering" of the given graph in linear time and then obtains a regular edge labeling from the canonical ordering in linear time.

The two works [12,22] were completely independent. Since there exists a linear-time algorithm to find the dual graph of a plane graph, algorithms in [1,8,11] can be used to find a rectangular drawing of a given plane graph and also Thomassen's characterization can be applied to VLSI floorplan problems.

A drawing in which each vertex is located at a grid point is called a *grid drawing*. It is a very challenging problem to draw a plane graph on a grid of the minimum size. In recent years, several works are devoted to this field [2,3,5,7,10,16,18,20]; for example, any plane graph with n vertices has a straight line drawing on a grid of area $WH \leqslant (n-1)^2$, where W is the width and H is the height of the grid. In the straight line drawing, edges are drawn by straight line segments, not necessarily by horizontal or vertical line segments.

In this paper we first give a new constructive proof of Thomassen's characterization, and then derive from the proof a simple algorithm to find a rectangular grid drawing of a plane graph in linear time. Our algorithm is completely different from those of [1,8,11], and is simple: we do not need to find any regular edge labeling or a canonical ordering, and our algorithm directly finds a rectangular grid drawing using a depth-first search. We also give upper bounds $WH \leqslant n^2/16$ on the the area and $W + H \leqslant n/2$ on the sum of width and height of rectangular grid drawings. These bounds are interesting in a sense that they are best possible and hold for any "compact" rectangular grid drawing of a plane graph.

The rest of the paper is organized as follows. Section 2 introduces some definitions and presents a characterization of plane graphs having rectangular drawings. Section 3 gives a rectangular drawing algorithm and shows that the algorithm takes linear time. Section 4 deals with rectangular grid drawings and upper bounds on the area and on the sum of width and height of the grid. Finally, Section 5 concludes with discussions and some open problems. An early version of the paper has been presented at [15].

2. Characterization of rectangular drawings

In this section we introduce some definitions and then prove a necessary and sufficient condition for a plane graph to have a rectangular drawing.

Let $G = (V, E)$ be a 2-connected graph with vertex set V and edge set E. We denote the set of vertices of G by $V(G)$, and the set of edges of G by $E(G)$. An edge joining vertices u and v is denoted by uv. The *union* $G = G' \cup G''$ of two graphs G' and G'' is a graph $G = (V(G') \cup$

$V(G''), E(G') \cup E(G'')$). A graph is *planar* if it can be embedded in the plane so that no two edges intersect geometrically except at a vertex to which they are both incident. A *plane graph* is a planar graph with a fixed embedding. A plane graph divides the plane into connected regions called *faces*. We regard the *contour* of a face as a *clockwise* cycle formed by the edges on the boundary of the face. We denote the contour of the outer face of graph G by $C_o(G)$ or simply C_o, and call C_o the *outer cycle* of G. In our input graph G all vertices have degree 3 except four vertices of degree 2 on $C_o(G)$. These four vertices divide C_o into four paths, the north path P_N, the east path P_E, the south path P_S, and the west path P_W. We will draw the north and south paths on two horizontal straight line segments and the east and west paths on two vertical line segments as depicted in Fig. 1. We thus fix the embedding of $C_o(G)$ as a rectangle. For $V' \subseteq V(G)$, $G - V'$ denotes a graph obtained from G by deleting all vertices in V' together with all edges incident to them. A C_o-*component* is either a K_2 whose vertices (but not edges) are in C_o or consists of a connected component of $G - V(C_o)$ together with the edges from that component to C_o and their ends [22]. Clearly the following lemma holds.

Lemma 2.1. *Let $H_1, H_2, \ldots, H_p$ be the C_o-components of a plane graph G, and let $G_i = C_o \cup H_i$, $1 \leqslant i \leqslant p$. Then G has a rectangular drawing with a fixed rectangular embedding of $C_o(G)$ if and only if G_i has a rectangular drawing with a fixed rectangular embedding of $C_o(G_i) = C_o(G)$ for each i, $1 \leqslant i \leqslant p$.*

In the remaining of this section, because of Lemma 2.1, we may assume that G has exactly one C_o-component as shown in Fig. 1(a).

For a cycle C in a plane graph G we denote by $G(C)$ the plane subgraph of G inside C (including C). An edge incident to a vertex on a cycle C and located outside C is called a *leg* of the cycle C. We have the following lemmas.

Lemma 2.2. *G has no rectangular drawing if the C_o-component has a cycle with less than four legs.*

Proof. If a cycle in the C_o-component has less than four legs, it is impossible to draw all inner faces, that are outside the cycle and whose contours contain edges of the cycle, as rectangles (see Fig. 3). □

A cycle C in G is *attached* to a path P if
(i) P does not contain any vertices in the proper inside of C, and
(ii) the intersection of C and P is a single subpath of P.
Let v_t be the starting vertex of the subpath, and let v_h be the ending vertex. We then call v_t the *tail vertex* of C for P, and v_h the *head vertex*. Denote by $Q_c(C)$ the path on C turning clockwise around C from v_t to v_h and by $Q_{cc}(C)$ the path on C turning counterclockwise around C from v_t to v_h. A leg of C is called a *clockwise leg* for P if it is incident to a vertex in $V(Q_c(C)) - \{v_t, v_h\}$. Denote by $n_c(C)$ the number of clockwise legs of C for P. Similarly we define a *counterclockwise leg* and denote by $n_{cc}(C)$ the number of counterclockwise legs of C for P. A cycle C attached to P is called a *clockwise cycle* if $Q_{cc}(C)$ is a subpath of P, and is called a *counterclockwise cycle* if $Q_c(C)$ is a subpath of P. A cycle C is called a *critical cycle* if either C is a clockwise cycle and $n_c(C) \leqslant 1$ or C is a counterclockwise cycle and $n_{cc}(C) \leqslant 1$. Fig. 4 illustrates a clockwise critical cycle. We have the following two lemmas.

Lemma 2.3. *G has no rectangular drawing if G has a critical cycle C attached to path P_N, P_E, P_S or P_W, except the outer cycle C_o.*

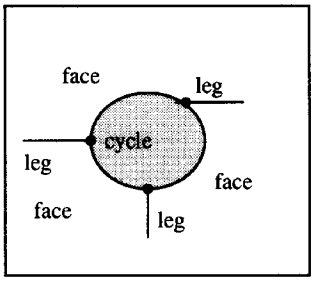

Fig. 3. Illustration for Lemma 2.2.

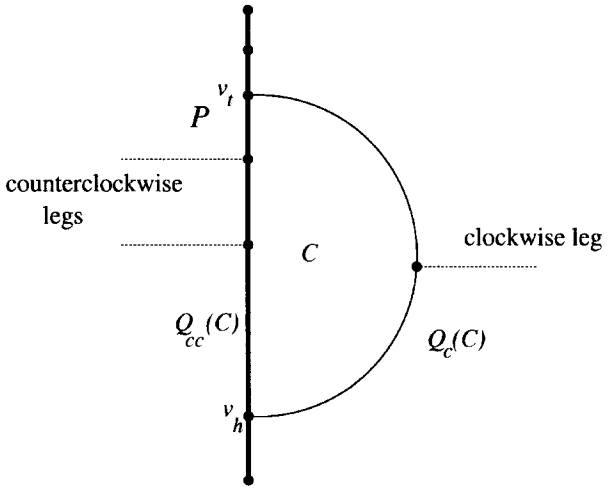

Fig. 4. Clockwise critical cycle C attached to path P.

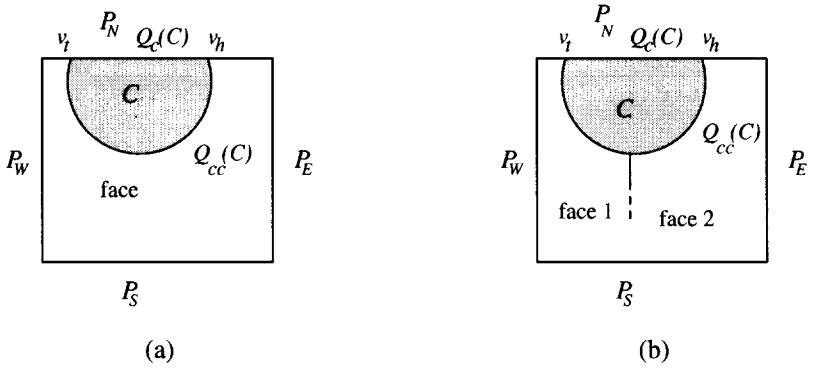

Fig. 5. Illustration for Lemma 2.3. (a) Case when $n_{cc}(C) = 0$. (b) Case when $n_{cc}(C) = 1$.

Proof. If G has a critical cycle C attached to path P_N, P_E, P_S or P_W, then it is impossible to draw all inner faces, that are outside C and whose contours contain edges of $Q_{cc}(C)$, as rectangles (see Fig. 5). □

Lemma 2.4. *G has no rectangular drawing if G has a cycle C with $n_{cc}(C) = 0$ attached to path $P = P_N + P_E$, $P_E + P_S$, $P_S + P_W$, or $P_W + P_N$, except the outer cycle C_o.*

Proof. It is impossible to draw the inner face, which is outside C and whose contour contains $Q_{cc}(C)$, as a rectangle (see Fig. 6). □

A C_o-component is called a *bad component* if it satisfies either Lemma 2.2 or Lemma 2.3 or Lemma 2.4. In particular, we call the C_o-component mentioned in Lemma 2.4 a *bad corner*. We now have the following theorem on the necessary and sufficient condition for a graph to have a rectangular drawing.

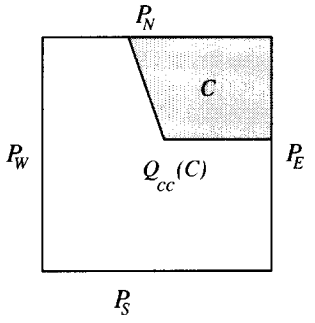

Fig. 6. Illustration for Lemma 2.4.

Theorem 2.5. *Let G be a plane graph such that all vertices have degree 3 except for the four corner vertices of degree 2 dividing the outer cycle C_o into four paths P_N, P_E, P_S and P_W. Then G has a rectangular drawing if and only if G has no bad component.*

Theorem 2.5 is essentially equivalent to Thomassen's characterization [22, Theorem 7.1], although the presentations are different. The proof of the necessity of Theorem 2.5 immediately follows from Lemmas 2.2–2.4. In the remaining of this section we give a constructive proof of the sufficiency, which will yield a linear time algorithm to find a rectangular grid drawing of G. We thus assume that G has no bad component.

Let $P_N = v_0v_1, v_1v_2, \ldots, v_{p-1}v_p$ and $P_S = u_0u_1, u_1u_2, \ldots, u_{q-1}u_q$. An *NS-path* is defined to be a path starting at a vertex v_i on P_N and ending at a vertex u_j on P_S without passing through any other vertex on C_o. An NS-path P divides graph G into two subgraphs G_W^P and G_E^P; G_W^P is the west part of G including P, and G_E^P is the east part of G including P. Drawing P as a straight line segment, we fix the embedding of $C_o(G_W^P)$ as a rectangle with the north path $P'_N = v_0v_1, v_1v_2, \ldots, v_{i-1}v_i$, the east path $P'_E = P$, the south path $P'_S = u_ju_{j+1}, u_{j+1}u_{j+2}, \ldots, u_{q-1}u_q$, and the west path $P'_W = P_W$. Similarly we fix the embedding of $C_o(G_E^P)$ as a rectangle. We say that P is an *NS-partitioning path* if neither G_W^P nor G_E^P has a bad component. Similarly we define *SN-, WE- and EW-partitioning paths*. If G has a partitioning path, say an NS-partitioning path P, then one can obtain a rectangular drawing of G by recursively finding rectangular drawings of G_W^P and G_E^P and patching them together along P.

An inner face of G is called a *boundary face* if its contour contains at least one edge of C_o. A *boundary path* is a maximal (directed) path on the contour of a boundary face connecting two vertices on C_o without passing through any edge on C_o. Note that the direction of a boundary path is the same as the contour of the face, and hence is clockwise. For $X, Y \in \{N, E, S, W\}$, a *boundary XY-path* is a boundary path starting at a vertex on P_X and ending at a vertex on P_Y. We have the following lemma.

Lemma 2.6. *Any boundary NS-, SN-, EW- or WE-path P of G is a partitioning path.*

Proof. One may assume that P is a boundary NS-path. Then P divides G into two subgraphs G_W^P and G_E^P. We shall show that neither G_W^P nor G_E^P has a bad component. Since P is a boundary NS-path, G_W^P is a cycle and hence G_W^P has no bad component. If G_E^P is also a cycle, then obviously G_E^P has no bad component. Thus one may assume that G_E^P is not a cycle. If G_E^P has a bad component mentioned

in Lemma 2.2, then it is also a bad component in G, a contradiction. If G_{E}^{P} has a bad component mentioned in Lemma 2.3, then G_{E}^{P} contains a critical cycle attached to P, and hence G contains a bad component as in Lemma 2.2, a contradiction. If G_{E}^{P} has a bad corner as in Lemma 2.4, then either G has a bad corner or G contains a bad component as in Lemma 2.3, a contradiction. Thus P is a partitioning path. □

Thus we may assume that G has none of boundary NS-, SN-, EW- and WE-paths. Then the C_{o}-component has at least one vertex on each of the paths $P_{\mathrm{N}}, P_{\mathrm{E}}, P_{\mathrm{S}}$ and P_{W}. In this case we find a pair of partitioning paths P_{c} and P_{cc} instead of a single partitioning path, and divide G into two or more subgraphs having no bad components by splitting G along P_{c} and P_{cc}. Both P_{c} and P_{cc} are NS-paths which have the same ends and do not cross each other in the plane although they share several edges. Thus, if $P_{\mathrm{c}} \neq P_{\mathrm{cc}}$, then $E(P_{\mathrm{c}}) \oplus E(P_{\mathrm{cc}}) = E(P_{\mathrm{c}}) \cup E(P_{\mathrm{cc}}) - E(P_{\mathrm{c}}) \cap E(P_{\mathrm{cc}})$ induces vertex-disjoint cycles $C_1, C_2, \ldots, C_k$, $k \geqslant 1$, as illustrated in Fig. 7. Thus P_{c} and P_{cc} share $k+1$ maximal subpaths $P_1, P_2, \ldots, P_{k+1}$. We assume that P_{c} turns around cycles $C_1, C_2, \ldots, C_k$ clockwise, and P_{cc} turns around them counterclockwise. We choose P_{c} and P_{cc} so that each cycle C_i has exactly four legs; assuming clockwise order, the first one is contained in P_i, the second one is a clockwise leg, the third one is contained in P_{i+1} and the fourth one is a counterclockwise leg. Thus G is divided into $G_{\mathrm{W}}^{P_{\mathrm{cc}}}$, $G_{\mathrm{E}}^{P_{\mathrm{c}}}$, $G(C_1), G(C_2), \ldots, G(C_k)$. In Fig. 1(a) P_{c} and P_{cc} are indicated by dotted lines. Fig. 1(b) depicts four subgraphs, $G_{\mathrm{W}}^{P_{\mathrm{cc}}}$, $G_{\mathrm{E}}^{P_{\mathrm{c}}}$, $G(C_1)$ and $G(C_2)$, obtained by splitting G in Fig. 1(a) along P_{c} and P_{cc}. We now have the following lemma.

Lemma 2.7. *Assume that G has no bad component and that a cycle C in the C_{o}-component has exactly four legs dividing C into four paths P'_{N}, P'_{E}, P'_{S} and P'_{W}. Then the subgraph $G(C)$ of G inside C has no bad component for any rectangular embedding of C fixed by P'_{N}, P'_{E}, P'_{S} and P'_{W}.*

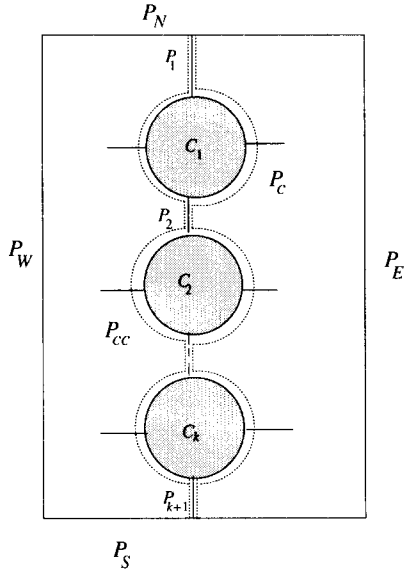

Fig. 7. Partition-pair P_{c} and P_{cc} indicated by dotted lines.

Proof. If $G(C)$ has a bad component for a rectangular embedding of C fixed by P'_{N}, P'_{E}, P'_{S} and P'_{W}, then it is also a bad component in G as mentioned in Lemma 2.2, a contradiction to the assumption that G has no bad component. □

By Lemma 2.7 we can assume that none of $G(C_1), G(C_2), \ldots, G(C_k)$ has a bad component for any fixed rectangular embeddings of $C_1, C_2, \ldots, C_k$. For each cycle C_i, $1 \leqslant i \leqslant k$, there are two alternative rectangular embeddings of C_i as illustrated in Fig. 8. Therefore there are 2^k different embeddings of cycles $C_1, C_2, \ldots, C_k$ where P_{c} and P_{cc} are embedded as alternating sequences of horizontal and vertical line segments (see Fig. 9). We can arbitrarily choose one of them, since none of $G(C_1), G(C_2), \ldots, G(C_k)$ has a bad component for any fixed rectangular embeddings of $C_1, C_2, \ldots, C_k$. Let G_1 be the graph obtained from $G_{\mathrm{W}}^{P_{\mathrm{cc}}}$ by contracting all edges of P_{cc} that are on the horizontal sides of rectangular embeddings of $C_1, C_2, \ldots, C_k$. (*Contracting an edge* $e = vw$ means removing it and identifying its ends v and w so that the resulting vertex is incident with those edges

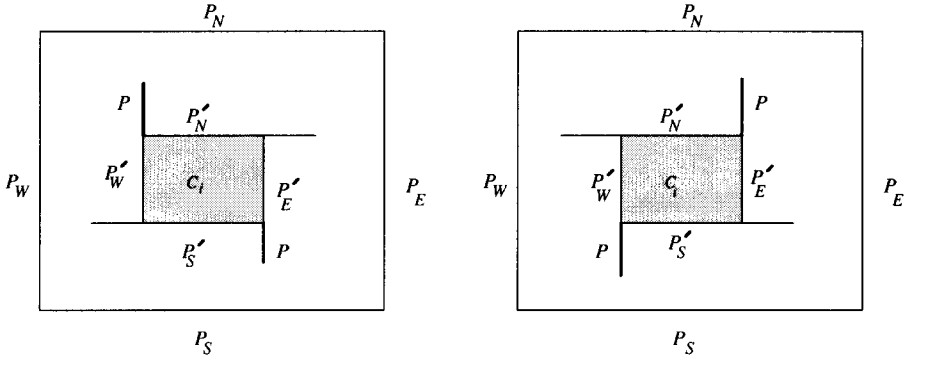

Fig. 8. Two alternative rectangular embeddings of cycle C_i.

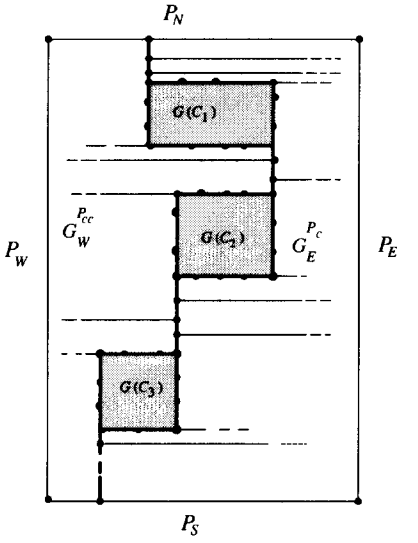

Fig. 9. An example of the embedding of a partition-pair.

(other than e) that were originally incident with v or w.) We denote by P'_{cc} the resulting path obtained from P_{cc} by the contraction above (see Fig. 1(c)). Then one can observe that if G_1 has a rectangular drawing with fixing the path P'_{cc} as a vertical straight line segment, then the rectangular drawing of G_1 can be easily modified to a drawing of $G_W^{P_{cc}}$ fitted in the area for $G_W^{P_{cc}}$ where P_{cc} is drawn as an alternating sequence of horizontal and vertical line segments. Let G_2 be the graph obtained from $G_E^{P_c}$ by contracting all edges of P_c that are on the horizontal sides of rectangular embeddings of $C_1, C_2, \ldots, C_k$ and let P'_c be the resulting path obtained from P_c by the contraction (see Fig. 1(d)). Then, if G_2 has a rectangular drawing, it can be easily modified to a drawing of $G_E^{P_c}$ fitted in the area for $G_E^{P_c}$ where P_c is drawn as an alternating sequence of horizontal and vertical line segments. Thus if we have drawings of graphs $G_W^{P_{cc}}, G_E^{P_c}, G(C_1), G(C_2), \ldots, G(C_k)$, then we can immediately patch them to get a rectangular drawing of G. One can observe that G_1 and G_2 have no bad components if and only if $G_W^{P_{cc}}$ and $G_E^{P_c}$ have no bad components when P_{cc} and P_c are fixed as straight line segments, respectively. Note that P'_{cc} and P'_c are obtained from P_{cc} and P_c, respectively, by contracting those edges which are on the horizontal sides of the rectangular embeddings of $C_1, C_2, \ldots, C_k$, and that all the vertices on these horizontal sides except the ends have degree 2 in $G_W^{P_{cc}}$ or $G_E^{P_c}$. We thus call P_c and P_{cc} a *pair of partitioning paths* or simply a *partition-pair* if neither $G_E^{P_c}$ nor $G_W^{P_{cc}}$ has a bad component. Especially when $P_c = P_{cc}$, it is a single partitioning path.

In Fig. 1(a), for the pair of partitioning paths P_c and P_{cc} indicated by dotted lines, there are two cycles C_1 and C_2, i.e., $k = 2$. Figs. 1(c) and (d) depict rectangular drawings of G_1 and G_2, and Fig. 1(b) depicts their modifications $G_W^{P_{cc}}$ and $G_E^{P_c}$ fitted in the areas with alternating sequences of horizontal and vertical line segments. Rectangular drawings of $G(C_1)$ and $G(C_2)$ are also depicted in Fig. 1(b). These can be patched into a rectangular drawing of G shown in Fig. 1(a).

Thus the problem is how to find a partition-pair efficiently. Our idea is to find a partition-pair from the "westmost NS-path" defined as follows. An NS-path P is *westmost* if

(1) P starts at v_1,

(2) P ends at u_{q-1}, and

(3) the number of edges in G_W^P is minimum.

The westmost NS-path is drawn in thick lines in Fig. 1(a). One can find the westmost NS-path by the "counterclockwise depth-first search" starting from v_1, that is, a depth-first search where the edge counterclockwise next to the currently visited edge is searched in each step.

The C_o-component may have cycles attached to the westmost NS-path P. Clearly all these cycles are clockwise attached to P. Either the proper insides of any two of the cycles are disjoint or one is contained in the other. A cycle among them is said to be *maximal* if its inside is not contained in the inside of any other cycle. Fig. 10 illustrates the hierarchical structure of the cycles, where the westmost path P is drawn on a vertical line and the insides of seven maximal critical cycles $C_{m1}, C_{m2}, \ldots, C_{m7}$ are shaded.

Lemma 2.8. *If G has no bad component and has none of boundary NS-, SN-, EW- and WE-paths, then G has a partition-pair P_c and P_{cc}.*

Proof. Assume that G has no bad component and has none of boundary NS-, SN-, EW- and WE-paths. Then we can find a partition-pair P_c and P_{cc} from the westmost NS-path $P = e_1, e_2, \ldots, e_j$ as follows.

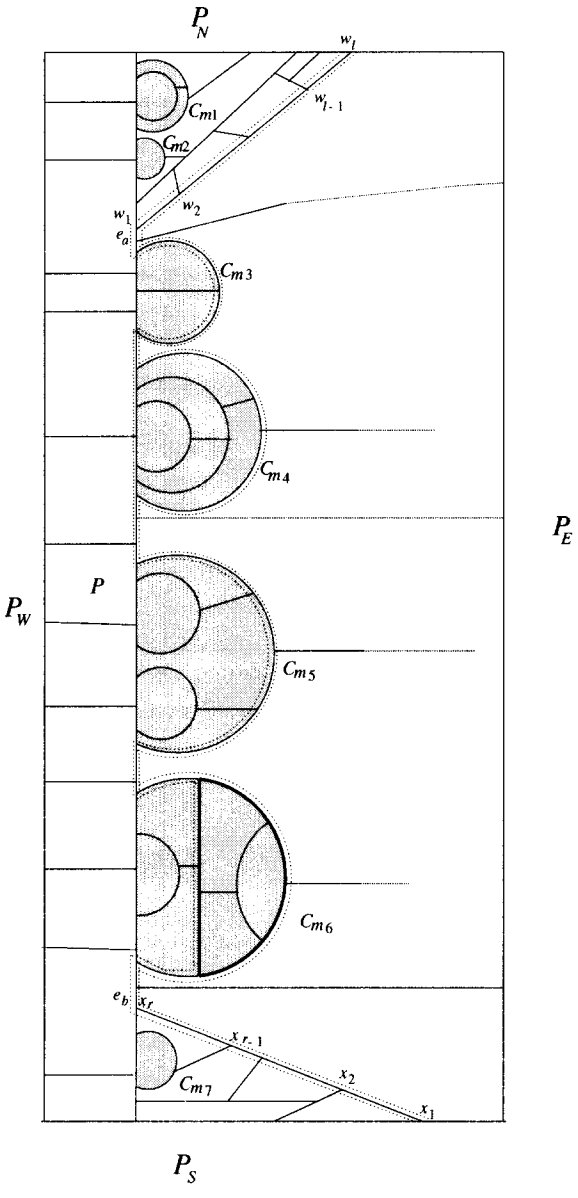

Fig. 10. Clockwise critical cycles attached to the westmost NS-path P.

Firstly, we find two paths P_{st} and P_{en}; P_{st} is the starting subpath shared by P_{c} and P_{cc}, and P_{en} is the ending subpath shared by P_{c} and P_{cc}. Let a be the largest index among $1, 2, \ldots, j$ such that $e_a \in E(P)$ is contained in a boundary NN- or EN-path, say $Q = \ldots, e_a, w_1w_2, \ldots, w_{l-1}w_l$, where $w_l \in V(P_{\mathrm{N}})$ (see Fig. 10). Choose $P_{\mathrm{st}} = w_lw_{l-1}, \ldots, w_2w_1, e_a$. Similarly, let b be the smallest index such that $e_b \in E(P)$ is contained in a boundary SS- or SE-path, say $R = x_1x_2, \ldots, x_{r-1}x_r, e_b, \ldots$, where $x_1 \in V(P_{\mathrm{S}})$. Choose $P_{\mathrm{en}} = e_b, x_rx_{r-1}, \ldots, x_2x_1$. Then $a < b$; otherwise, there would exist a boundary SN-path. Furthermore, neither $G_{\mathrm{E}}^{P'}$ nor $G_{\mathrm{W}}^{P'}$ has a bad corner for any NS-path P' whose

starting subpath is P_{st} and ending subpath is P_{en}; otherwise, a contradiction either to the selection of P_{st} and P_{en} or to the assumption that G has no bad component would occur.

Secondly, if an edge among $e_a, e_{a+1}, \ldots, e_b$ on P is not contained in any maximal critical cycle C attached to P, then we let both P_c and P_{cc} pass through it, that is, we include it in a common subpath $P_1, P_2, \ldots,$ or P_{k+1}. (We will give the detail how to find maximal critical cycles later in Section 3.)

Lastly, we choose subpaths of P_c and P_{cc} for each of the clockwise maximal critical cycles C attached to the subpath of P connecting e_a and e_b, as follows. (In Fig. 10 C_{m3}, C_{m4}, C_{m5} and C_{m6} are these maximal cycles.) Since $n_c(C) \leqslant 1$, we have the following three cases.

Case 1: $n_c(C) = 0$.

In this case $n_{cc}(C) \geqslant 2$; otherwise, G would have a bad component, contrary to the assumption. We let both P_c and P_{cc} pass through $Q_c(C)$. (In Fig. 10 $n_c(C_{m3}) = 0$ and $n_{cc}(C_{m3}) = 2$.)

Case 2: $n_c(C) = 1$ and $n_{cc}(C) \leqslant 1$.

In this case $n_{cc}(C) = 1$; otherwise, G would have a bad component. Thus C has exactly four legs. We let path P_c pass through $Q_c(C)$ and let P_{cc} pass through $Q_{cc}(C)$. Thus C is one of the cycles induced by $E(P_c) \oplus E(P_{cc})$. (In Fig. 10 $n_c(C_{m4}) = 1$ and $n_{cc}(C_{m4}) = 1$.)

Case 3: $n_c(C) = 1$ and $n_{cc}(C) \geqslant 2$.

In this case there are two subcases. In the first subcase where $G(C)$ has no counterclockwise critical cycle attached to $Q_c(C)$, we let both P_c and P_{cc} pass through $Q_c(C)$. (In Fig. 10 $n_c(C_{m5}) = 1$ and $n_{cc}(C_{m5}) = 2$ and $G(C_{m5})$ has no counterclockwise critical cycle attached to $Q_c(C_{m5})$.) Consider the second subcase in which $G(C)$ has one or more counterclockwise critical cycles attached to $Q_c(C)$. In this case $G(C)$ has exactly one maximal counterclockwise critical cycle C' attached to $Q_c(C)$; if $G(C)$ has two or more maximal critical cycles attached to $Q_c(C)$, then there would exist a cycle with three legs, contrary to the assumption. Furthermore, $n_{cc}(C') = 1$; otherwise, $n_{cc}(C') = 0$ and hence C' would have exactly three legs, contradicting the assumption. Since $n_{cc}(C') = 1$, C' has exactly four legs, and hence we let both P_c and P_{cc} pass through edges in $E(Q_c(C)) - E(C')$, let P_c pass through $Q_c(C')$, and let P_{cc} pass through $Q_{cc}(C')$. Thus C' is one of the cycles induced by $E(P_c) \oplus E(P_{cc})$. (In Fig. 10 $n_c(C_{m6}) = 1, n_{cc}(C_{m6}) = 2$, and $G(C_{m6})$ has exactly one maximal counterclockwise critical cycle C' attached to $Q_c(C_{m6})$ which is drawn in thick lines.)

If we choose P_c and P_{cc} as above, then clearly each of the cycles induced by $E(P_c) \oplus E(P_{cc})$ has exactly four legs, two of which are shared by P_c and P_{cc}. Furthermore, one can observe that both $G_E^{P_c}$ and $G_W^{P_{cc}}$ have no bad component. Therefore P_c and P_{cc} are a pair of partitioning paths. (In Fig. 10 $E(P_c) \oplus E(P_{cc})$ induces two cycles. In Fig. 1 Case 2 and the second subcase of Case 3 have occurred.) □

Using Lemmas 2.6–2.8, one can recursively find a rectangular drawing of a given plane graph G if it has no bad component. Thus we have constructively proved the sufficiency of Theorem 2.5.

3. Rectangular drawing algorithm

In this section, we give the following algorithm DRAW-GRAPH to find a rectangular drawing of a plane graph G if it exists. Our algorithm can be easily modified to check the existence of a rectangular drawing of a plane graph G. As the output of the algorithm we take the directions (vertical

or horizontal) of edges of G, although we can take as the output either the directions or the real-valued coordinates of vertices in a rectangular drawing. From the directions one can decide the integer coordinates of vertices as we show later in the succeeding section.

We treat each C_o-component independently as in Lemma 2.1. If there exists a boundary NS-, SN-, WE- or EW-path, we choose it as a partitioning path. Otherwise, we find a partition-pair P_c and P_{cc} from the westmost NS-path, and then recurse to the subgraphs divided by P_c and P_{cc}.

Algorithm DRAW-GRAPH(G)
begin
1. draw the contour $C_o(G)$ of the outer face of G as a rectangle by two horizontal line segments P_N and P_S and two vertical line segments P_E and P_W;
   {the directions of edges on C_o are decided}
2. find all C_o-components $H_1, H_2, \ldots, H_p$;
3. **for** each C_o-component H_i **do**
   begin
4. $G_i = C_o \cup H_i$; {G_i is the union of graphs C_o and H_i}
5. DRAW(H_i, G_i)
   end
end.
Procedure DRAW(H, G)
{H is the C_o-component of graph G}
begin
1. **if** G has a boundary NS-, SN-, EW- or WE-path P
   {P is a partitioning path}
   then
   begin
   assume without loss of generality that P is a boundary NS-path;
2. draw all edges of P on a vertical line;
   {the directions of edges of P are decided to be vertical}
3. **if** $|E(P)| \geqslant 2$ **then**
   begin
   let $F_1, F_2, \ldots, F_q$ be the C_o-components of G_E^P for the fixed rectangular embedding of cycle $C_o(G_E^P)$; {G_W^P is a cycle}
4. **for** each F_i **do** DRAW($F_i, C_o(G_E^P) \cup F_i$)
   end
   end
5. **else** {G has none of boundary NS-, SN-, EW- and WE-paths}
   begin
6. find the westmost NS-path P;
7. find a partition-pair P_c and P_{cc} from P as in the proof of Lemma 2.8;
8. **if** $P_c = P_{cc}$ **then**
   begin
9. draw all edges of P_c on a vertical line segment;
   let $G_1 = G_W^{P_c}$ and $G_2 = G_E^{P_c}$ be two resulting subgraphs with fixed rectangular em-

```
            beddings of cycles C_o(G_W^{P_c}) and C_o(G_E^{P_c});
10.         for each G_i do
              begin
                let F_1, F_2, ..., F_q be the C_o-components of G_i;
11.             for each F_j do DRAW(F_j, C_o(G_i) ∪ F_j)
              end
          end
12.     else
          begin
13.       draw all edges of P_c and P_cc on alternating sequences of horizontal and vertical line
            segments as in Fig. 9;
              {the directions of all edges of P_c and P_cc are decided}
            let G_1 be the graph obtained from G_W^{P_cc} by contracting all edges of P_cc that are on
            the horizontal sides of rectangular embeddings of C_1, C_2, ..., C_k;
            let G_2 be the graph obtained from G_E^{P_c} by contracting all edges of P_c that are on the
            horizontal sides of rectangular embeddings of C_1, C_2, ..., C_k;
            let G_3 = G(C_1), ..., G_{k+2} = G(C_k) be the subgraphs with fixed rectangular embed-
            dings of cycles C_1, ..., C_k;
14.         for each G_i do
              begin
                let F_1, F_2, ..., F_q be the C_o-components of G_i;
15.             for each F_j do DRAW(F_j, C_o(G_i) ∪ F_j)
              end
          end
      end
end
```

We now show that the algorithm DRAW-GRAPH(G) takes linear time.

We find all C_o-components of G, and for each C_o-component we find boundary NS-, SN-, EW- and WE-paths if they exist. We do this by traversing all boundary paths of G using the counterclockwise depth-first search. During the traversal every edge on a boundary path is marked according to the boundary path, and each boundary path gets a label, NN, NE, ..., WW, according to the location of their starting and ending vertices on P_N, P_E, P_S and P_W. Therefore boundary NS-, SN-, EW- and WE-paths, if they exist, can be readily found from the labels of the boundary paths in constant time.

We then need to find the westmost NS-path P if no boundary NS-, SN-, EW- or WE-paths exists. We find P by traversing the boundary paths with ends on P_W using the counterclockwise depth-first search. During the traversal, we can find all edges that are on P and on a boundary NN-, EN-, SS- or SE-path by checking the labels of boundary paths on which edges of P lie. Thus we can find P_{st} and P_{en} as mentioned in the proof of Lemma 2.8. Traversing the contours of all faces clockwise attached to the subpath of P connecting e_a and e_b, we detect the clockwise critical cycles attached to the subpath if they exist. An edge, which is not incident to a vertex on P and is traversed twice during this traversal, is detected as the leg of a clockwise critical cycle. One can observe that the head vertices and the tail vertices of all the critical cycles attached to P obey the so-called parenthesis rule. Therefore, considering the parenthesis structures of the found critical cy-

cles attached to P, we can find the maximal critical cycles attached to P. From the found maximal critical cycles we find a partition-pair P_c and P_{cc} as mentioned in the proof of Lemma 2.8. One can do this by traversing the following edges a constant number of times: (i) the edges on P_c and P_{cc}, (ii) the edges on the contour of the faces clockwise attached to the subpath of P connecting e_a and e_b, and (iii) the edges on boundary paths newly created in the graphs divided by P_c and P_{cc}.

After finding a partitioning path or a partition-pair, we give labels to the newly created boundary paths by traversing them. The labels of some old boundary paths are updated for the newly found partitioning path or partition-pair. Clearly this can be done by traversing the respective boundary paths only once.

A problem arises if a subpath of the westmost NS-path P, which is neither on P_c nor on P_{cc}, is chosen as the westmost NS-path P' in a later recursive stage. If we again traversed the contour of the faces attached to P' as mentioned before, then the time complexity of the algorithm would not be bounded by linear time. To overcome this difficulty, we keep the following information for later use when P is first constructed:

(i) a list of all edges $e_i \in E(P)$ contained in boundary NN- and EN-paths;
(ii) a list of all edges $e_i \in E(P)$ contained in boundary SS- and SE-paths;
(iii) an array of length $|V|$ containing marks indicating whether the vertex corresponding to each element is a head or a tail vertex of a clockwise critical cycle C attached to P and whether $n_{cc}(C) = 1$ or $n_{cc}(C) > 1$.

We use lists of (i) and (ii) to find P_{st} and P_{en} directly in later recursive stages. Marks of vertices in (iii) indicate the existence of critical cycles attached to P'. Since we select the westmost NS-path P' in a later recursive stage, we need not to find these critical cycles again.

Throughout the execution of the algorithm, every face of G become a boundary face and then will never become non-boundary face. Hence each face is traversed by a constant number of times. Therefore the algorithm runs in linear time.

Theorem 3.1. *The algorithm DRAW-GRAPH finds a rectangular drawing of a given plane graph in linear time if it exists.*

4. Rectangular grid drawings

In this section we first show how to find a rectangular grid drawing from a rectangular drawing of a given plane graph G in linear time, and then give upper bounds on the area and the sum of width and height of the grid.

The algorithm DRAW-GRAPH(G) in the preceding section only finds the directions of all edges in G. From the directions the integer coordinates of vertices in G can be determined in linear time [18,20]. To make this paper self-contained we give an alternative simple strategy as follows.

We now give a method of determining y-coordinates of the vertices in G; x-coordinates can be determined similarly. For each vertex v in G we will assign an integer temp(v) as a temporary value of the y-coordinate of v. The rectangular drawing of G is composed of some maximal horizontal and vertical line segments. For each maximal horizontal line segment L, we will assign an integer $y(L)$ as the y-coordinate for every vertex v on L. There are two cases: either v has a neighbor u located

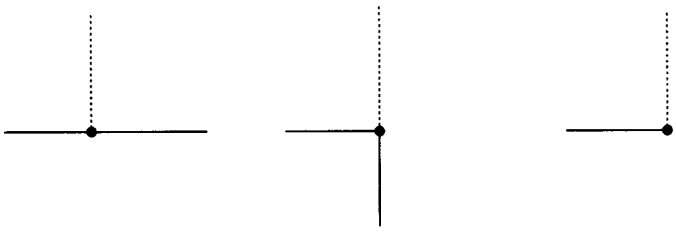

Fig. 11. Illustration of three types of edges.

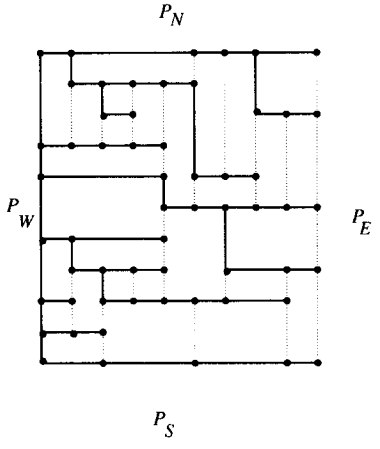

Fig. 12. Illustration of T_y by thick lines.

below v or v has no neighbor u located below v. For the former case, we set $\text{temp}(v) = y(L') + 1$ where L' is the maximal horizontal line segment containing vertex u. For the latter case, we set $\text{temp}(v) = 0$. We then set $y(L) = \max_v\{\text{temp}(v)\}$ where the maximum is taken over all vertices v on L. Clearly $y(P_S) = 0$ since $\text{temp}(v) = 0$ for all vertices v on P_S. Consider a graph T_y obtained from G by deleting all upward vertical edges of three types indicated by dotted lines in Fig. 11. T_y is a spanning tree of G, because the contour of each face of G contains at least one vertical edge of the three types and the upward vertical edge starting from the west end of each maximal horizontal line segment is not deleted. (T_y for the graph G in Fig. 1(a) is drawn in thick lines in Fig. 12.) One can easily compute $y(L)$ for all maximal horizontal line segments L from bottom to top using the counterclockwise depth-first search on T_y starting from the downward edge incident to the north-west corner vertex of degree two.

Thus we have shown that a rectangular grid drawing can be obtained in linear time.

Let the coordinate of the south-west corner be $(0, 0)$, and let that of the north-east corner be (W, H). Then our grid drawing is "compact" in a sense that there is at least one vertical line segment of x-coordinate i for each i, $0 \leqslant i \leqslant W$, and there is at least one horizontal line segment of y-coordinate j for each j, $0 \leqslant j \leqslant H$. We have the following theorem on the sizes of a compact rectangular grid drawing.

Theorem 4.1. *The sizes of any compact rectangular grid drawing D of G satisfy $W + H \leqslant n/2$ and $WH \leqslant n^2/16$.*

Proof. Let l be the number of maximal horizontal and vertical line segments in D. Each of the segments has exactly two ends. Each of the vertices except the four corner ones is an end of exactly one of the $l-4$ maximal line segments other than P_N, P_E, P_S and P_W. Therefore we have

$$n-4=2(l-4)$$

and hence

$$l-2=\frac{n}{2}. \tag{1}$$

Let l_h be the number of maximal horizontal line segments and l_v the number of maximal vertical line segments in D. Since D is compact, we have

$$H \leqslant l_h - 1 \tag{2}$$

and

$$W \leqslant l_v - 1. \tag{3}$$

By using (1)–(3), we obtain

$$W+H \leqslant l_v + l_h - 2 = l - 2 = \frac{n}{2}.$$

This relation immediately implies the bound on area: $WH \leqslant n^2/16$. □

The bounds above are tight, because there are an infinite number of examples attaining the bounds, as one in Fig. 13.

Several results have been published on the bounds on width and height of *orthogonal grid drawings* of graphs with the maximum degree $\leqslant 4$ where edges are drawn as alternating sequences of horizontal and vertical line segments with bends and faces are not necessarily drawn as rectangles [2,17]. After publishing the preliminary version of our paper [15], Biedl [3] obtained a bound

$$W+H \leqslant b+2n-m-2$$

for any orthogonal grid drawing of a plane graph with n vertices, m edges, and b bends. This result also implies the same bound $W+H \leqslant n/2$ for rectangular grid drawings of our input graphs.

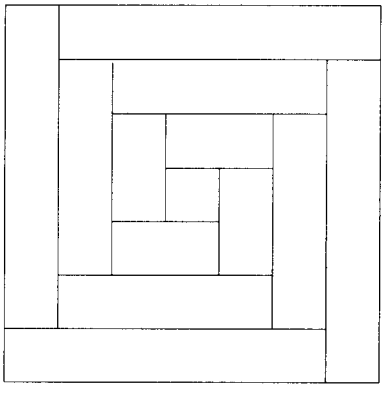

Fig. 13. An example of a rectangular grid drawing attaining the upper bounds.

5. Conclusion

In this paper we presented a simple linear-time algorithm to find a rectangular grid drawing of a plane graph and also gave upper bounds on the area and the sum of width and height of the grid. The bounds are tight and best possible.

Our algorithm always finds a partition-pair from the *westmost NS-path*. However, any of the four sides of the rectangular embedding of C_o can be considered as P_N. Therefore we may generate different rectangular drawings if we apply our algorithm after rotating G by 90°, 180° or 270°. Furthermore, there are 2^k different rectangular embeddings of $C_1, C_2, \ldots, C_k$ when $E(P_c) \oplus E(P_{cc})$ induces cycles $C_1, C_2, \ldots, C_k$ for a partition-pair P_c and P_{cc}. Therefore we can generate many rectangular drawings of G by rotating G and by choosing different combinations of embeddings of induced cycles for partition-pairs. Such an algorithm is useful for finding a suitable rectangular drawing for an efficient VLSI-floorplanning [21].

Although we consider that our input graph has exactly four vertices of degree 2 on C_o, our technique can handle the case where there are more than four vertices of degree 2 on C_o. In this case, we select four suitable vertices of degree 2 as the four corner vertices of a rectangular embedding of C_o, and then apply our algorithm.

Our work raises several interesting open problems:

(1) What is the necessary and sufficient condition to have a rectangular drawing of a plane graph with vertices of degree less than or equal to 4?

(2) What is the complexity of an optimal parallel algorithm for rectangular grid drawings [9]?

Acknowledgements

We wish to thank the three anonymous referees for their valuable comments and suggestions for improving the presentation of the paper.

References

[1] J. Bhasker, S. Sahni, A linear algorithm to find a rectangular dual of a planar triangulated graph, Algorithmica 3 (1988) 247–278.

[2] T.C. Biedl, New lower bounds for orthogonal graph drawings, in: Proc. Graph Drawing '95, Lecture Notes in Computer Science 1027, 1996, pp. 28–39.

[3] T.C. Biedl, Optimal orthogonal drawings of triconnected plane graphs, in: Proc. Scandinavian Workshop on Algorithm Theory, SWAT'96, Lecture Notes in Computer Science 1097, 1996, pp. 333–344.

[4] N. Chiba, K. Onoguchi, T. Nishizeki, Drawing planar graphs nicely, Acta Informatica 22 (1985) 187–201.

[5] M. Chrobak, S. Nakano, Minimum-width grid drawings of plane graphs, Technical Report, UCR-CS-94-5, Department of Computer Science, University of California at Riverside, 1994. Also in: Proc. Graph Drawing '94, Lecture Notes in Computer Science 894, pp. 104–110.

[6] G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis, Algorithms for drawing graphs: an annotated bibliography, Computational Geometry: Theory and Applications 4 (5) (1994) 235–282.

[7] H. de Fraysseix, J. Pach, R. Pollack, How to draw a planar graph on a grid, Combinatorica 10 (1990) 41–51.

[8] X. He, On finding the rectangular duals of planar triangulated graphs, SIAM J. Comput. 22 (6) (1993) 1218–1226.
[9] X. He, An efficient parallel algorithm for finding rectangular duals of plane triangulated graphs, Algorithmica 13 (1995) 553–572.
[10] X. He, Grid embedding of 4-connected plane graphs, Discrete Comput. Geom. 17 (1997) 339–358.
[11] G. Kant, X. He, Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems, Theor. Comput. Sci. 172 (1997) 175–193.
[12] K. Kozminski, E. Kinnen, An algorithm for finding a rectangular dual of a planar graph for use in area planning for VLSI integrated circuits, in: Proc. 21st DAC, Albuquerque, June 1984, pp. 655–656.
[13] T. Lengauer, Combinatorial Algorithms for Integrated Circuit Layout, Wiley, Chichester, 1990.
[14] T. Nishizeki, N. Chiba, Planar Graphs: Theory and Algorithms, North-Holland, Amsterdam, 1988.
[15] M.S. Rahman, S. Nakano, T. Nishizeki, Rectangular grid drawings of plane graphs, in: Computing and Combinatorics, Proc. 2nd Annual International Conference, COCOON'96, Lecture Notes in Computer Science 1090, 1996, pp. 92–105.
[16] W. Schnyder, Embedding planar graphs in the grid, in: Proc. 1st ACM–SIAM Symposium on Discrete Algorithms, San Francisco, 1990, pp. 138–147.
[17] J.A. Storer, On minimal node-cost planar embeddings, Networks 14 (1984) 181–212.
[18] R. Tamassia, On embedding a graph in the grid with the minimum number of bends, SIAM J. Comput. 16 (3) (1987) 421–444.
[19] R. Tamassia, G. Di Battista, C. Batini, Automatic graph drawing and readability of diagrams, IEEE Trans. Systems Man Cybernet. 18 (1988) 61–79.
[20] R. Tamassia, I.G. Tollis, Planar grid embedding in linear time, IEEE Trans. Circuits Systems 36 (9) (1989) 1230–1234.
[21] K. Tani, S. Tsukiyama, S. Shinoda, I. Shirakawa, On area-efficient drawings of rectangular duals for VLSI floor-plan, Math. Programming 52 (1991) 29–43.
[22] C. Thomassen, Plane representations of graphs, in: J.A. Bondy, U.S.R. Murty (Eds.), Progress in Graph Theory, Academic Press, Canada, 1984, pp. 43–69.